FOCUS OF ATTENTION IN AN ACTIVITY-BASED SCHEDULER

Norman Sadeh and Mark S. Fox

Computer Science Department and Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Earlier research in job shop scheduling has demonstrated the advantages of opportunistically combining order-based and resource-based scheduling techniques. In this paper we investigate an even more flexible approach where each activity is considered a decision point by itself. Heuristics to opportunistically select the next decision point on which to focus attention (i.e., *variable ordering* heuristics) and the next decision to be tried at this point (i.e., *value ordering* heuristics) are described that probabilistically account for both activity precedence and resource requirement interactions. Preliminary experimental results indicate that the variable ordering heuristic greatly increases search efficiency. While least constraining value ordering heuristics have been advocated in the literature [7], our experimental results suggest that other value ordering heuristics combined with our variable-ordering heuristic can produce much better schedules without significantly increasing search.

1. Introduction

We are concerned with the issue of how to opportunistically focus an incremental job shop scheduler's attention on the most critical decision points (*variable ordering* heuristics) and the most promising decisions at these points (*value ordering* heuristics) in order to reduce search and improve the quality of the resulting schedule.

So called order-based and resource-based scheduling techniques have been at the origin of several incremental scheduling algorithms. In an order-based approach, each order is considered a single decision point, i.e. orders are prioritized and scheduled one by one. In a resource-based approach, resources are rated according to their projected levels of demand. Resources are then scheduled one by one, starting with the ones that have the highest demand. Order-based scheduling has proven to be a viable paradigm in problems where slack (i.e. temporal precedence interactions) is the dominating factor. On the other hand, resource-based scheduling is likely to perform better in situations where resource contention (i.e. resource requirement interactions) is critical. Neither approach is perfect. Indeed a lot of real life scheduling problems contain a mix of critical orders and critical resources. In the past few years it has become clear that in order to perform well in a wider class of problems, schedulers need the ability to opportunistically switch from one approach to the other. The OPIS [15, 12, 16] scheduler was the first scheduler to combine both approaches. OPIS uses demand thresholds to identify bottleneck resources. Typically, when a bottleneck resource is detected, all activities requiring that resource and that have not yet been scheduled will be scheduled. When all bottleneck resources have been scheduled, OPIS switches to order-based scheduling. While in order scheduling mode, OPIS may detect the appearance of new bottleneck resources and switch back to its resource scheduling mode. Even such an approach has its shortcomings as the criticalities of the activities requiring a bottleneck resource or belonging to a critical order are not homogeneous, i.e. some of these activities may not be that critical. This consideration led us to the investigation of a new scheduling framework where the decision points are no longer entire resources or entire orders but instead where each activity is a decision point in its own right. Within

this framework activity criticality is no longer determined via a sole bottleneck resource or via the sole order to which it belongs. Instead measures of activity criticality account for both temporal precedence interactions (i.e., so-called *intra-order* interactions [15]) and resource requirement interactions (i.e., so-called *inter-order* interactions). By simultaneously accounting for both types of interactions the approach is expected to opportunistically combine advantages of both order-based and resource-based scheduling techniques.

In this paper, we study one variable-ordering heuristic and three value-ordering heuristics for activity-based scheduling. A preliminary set of 38 scheduling problems was used to compare the performances of these heuristics. The experiments clearly indicate that the variable-ordering heuristic significantly reduces search. The comparison of the value-ordering heuristics when combined with the variable ordering heuristic suggests that a least constraining value ordering heuristic is not the only viable approach to maintain the amount of search at an acceptable level. Indeed some other heuristics produced much better schedules without significantly increasing search.

In the next section we describe our model of the job-shop scheduling problem. The following section gives an overview of the activity-based approach to scheduling that we are investigating, and introduces a probabilistic model to account for both intra-order and inter-order interactions. Section 4 presents a variable-ordering heuristic based on this probabilistic model. In section 5 we present three value-ordering heuristics: a least constraining heuristic, a hill-climbing heuristic, and an intermediate heuristic where values are rated according to the probability that they will remain available and that no better values will remain available. Preliminary experimental results are reported in section 6. Section 7 discusses these results. Section 8 contains some concluding remarks.

2. The Model

Formally, we will say that we have a set of N jobs (i.e. orders) to schedule. Each job has a predefined process plan that specifies a partial ordering among the activities (i.e. operations) to be scheduled. Each activity A_k $(1 \le k \le n)$ may require one or several resources R_{ki} $(1 \le i \le p_k)$, for each of which there may be several alternatives R_{kij} $(1 \le j \le q_{ki})$[1]. We will use st_k, et_k, and du_k to respectively denote A_k's start time, end time, and duration.

We view the scheduling problem as a constraint satisfaction problem (CSP).

The variables of the problem are the activity start times, the resources allocated to each activity, and possibly the durations of some activities. An activity's end time is defined as the sum of its start time and duration. Each variable has a bounded (finite or infinite) set of admissible values. For instance, the start time of an activity is always restricted at one end by the order release date and at the other end by the order latest acceptable completion time[2] according to the durations of the activities that precede/follow the activity in the process plan.

We differentiate between two classes of constraints: activity precedence constraints and resource capacity constraints. The activity precedence constraints are the ones defined by the process plans. Our model [14] accounts for all 13 of Allen's temporal constraints [1]. Capacity constraints restrict the number of reservations of a resource over any time interval to the capacity of that resource. For the sake of simplicity, we will assume in this paper that all resources are of unary capacity.

Additionally our model allows for preferences on activity start times and durations as well as on the resources that activities can use. Preferences are modeled with utility functions. These functions map each variable's possible values onto utilities ranging between 0 and 1. Preferences on activity start times and durations allow for the

[1]It is important to keep in mind that several activities may require the same resource. For instance if two activities A_1 and A_2 both require a unique resource which has to be R_1, we have $R_{111} = R_{211} = R_1$.

[2]This is not necessarily the order's due date.

representation of organizational goals such as reducing order tardiness, or reducing work-in-process (WIP) [3, 14]. Resource preferences are very useful to differentiate between functionally equivalent resources with different characteristics (e.g. difference in accuracy). In this paper we will assume that the sum of the utility functions defines a (separable) objective function to be maximized.

3. The Approach

3.1. An Activity-based Scheduler

In an activity-based approach, each activity is treated as an aggregate variable, or decision point, that comprises the activity's start time, its resources, and possibly its duration. The schedule is built incrementally by iteratively selecting an activity to be scheduled and a reservation for that activity (i.e. start time, resources and possibly duration). Every time a new activity is scheduled, new constraints are added to the initial scheduling problem, and propagated. If an inconsistency is detected during propagation, the system backtracks. The process stops either when all activities have been successfully scheduled or when all possible alternatives have been tried without success.

The efficiency of such an incremental approach critically relies on the order in which activities are scheduled and on the order in which possible reservations are tried for each activity. Indeed, because job-shop scheduling is NP-hard, search for a schedule may require exponential time in the worst case. Both empirical and analytical studies of constraint satisfaction problems reported in [6, 5, 13, 9, 10, 11, 17] indicate however that, on the average, search can significantly be reduced if always focused on the most critical decision points and the most promising decisions at these points. Such techniques are often referred to [2] as variable and value ordering heuristics.

In this paper we assume that critical activities are the ones whose good (overall) reservations are most likely to become unavailable if one were to start scheduling other activities first. In general reservations may become unavailable because of operation precedence constraints, because of resource capacity constraints, or because of combinations of both types of constraints. Clearly criticality measures are probabilistic in nature, as their computations require probabilistic assumptions on the values that will be assigned later on to each variable (i.e. the reservations that will later on be assigned to each unscheduled activity). In the next subsection we introduce a probabilistic framework that accounts for the interactions of start time, duration and resource preferences induced by both activity precedence and resource capacity constraints. We will use this model throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based scheduling.

3.2. A Probabilistic Framework to Account for Constraint Interactions

In this subsection we outline[3] a probabilistic model that we will use throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based job-shop scheduling. We justify the model by its ability to account for both intra-order and inter-order interactions and by its relatively low computational requirements. For the sake of simplicity, the formulas presented in this paper assume fixed duration activities. Similar formulas can be deduced when dealing with variable duration activities.

In our model a priori probability distributions are assumed for the possible start times and resources of each unscheduled activity. These probabilities are then refined to account for the interactions induced by the problem constraints (i.e. both intra-order and inter-order interactions). Finally the results of this propagation process are combined to identify critical activities and promising reservations for these activities. In their simplest form the a priori probability distributions are uniform. This amounts to assuming that, a priori, all possible reservations are

[3] A more detailed description can be found in [14].

equally probable. A slightly more sophisticated model consists in *biasing* the a priori distributions towards good values as defined by the utility functions [14]. Such biased distributions are expected to account better for value ordering heuristics that give more attention to higher utility values.

Once the a priori distributions have been built, they can be refined to account for the interactions of the problem constraints. In our model, the propagation is performed in two steps. The probability distributions are first propagated within each order, thereby accounting for intra-order interactions, and then across orders to account for inter-order interactions. Accounting simultaneously for both types of interactions seems indeed very difficult as much from a theoretical point of view as from a purely computational point of view. As a matter of fact the number of ways in which a set of activities can interact is combinatorial in the number of these activities[4]. Instead, by separately accounting for intra-order and inter-order interactions, one greatly reduces the amount of computation to be performed. The propagation results can always be further refined by iterating the propagation an arbitrary number of times.

Concretely, once the a priori distributions have been generated, our propagation process involves the following two steps:

1.

   a. The a priori start time probability distributions are refined to account for activity precedence constraints. The resulting (a posteriori) probability distributions associate to the possible start times of each activity the probability that these start times will be tried by the scheduler and will not result in the violation of an activity precedence constraint. These a posteriori start time distributions can be normalized to express that each activity will occur exactly once, and hence will start exactly once.

   b. For each resource requirement R_{ki} of each activity A_k, and for each resource alternative R_{kij} to fulfill R_{ki}, we compute the probabilistic demand D_{kij} of A_k for R_{kij} as a function of time. This probability is obtained using A_k's normalized a posteriori start time distribution and the a priori probability that A_k uses R_{kij} to fulfill its requirement R_{ki}. Hence $D_{kij}(t)$ represents the probabilistic contribution of A_k to the demand for R_{kij} at time t, if activity reservations were only checked for consistency with respect to the activity precedence constraints. Later on we will refer to $D_{kij}(t)$ as A_k's (probabilistic) individual demand for R_{kij} at time t.

2. Finally the individual demand densities of all activities are aggregated (i.e. summed at each point in time) to reflect the probabilistic demand for each resource in function of time. The resulting aggregate demand densities may get larger than one over some intervals of time, as the individual demand densities from which they originate have not been checked for consistency with respect to the capacity constraints. High demand for a resource over some time interval indicates a critical resource/time interval pair, which requires prompt attention from the scheduler. This is the basis of the variable-ordering heuristic presented in this paper.

More precise probabilities may be obtained by iterating the propagation process. One way to do so consists in computing for each possible activity reservation the probability that this reservation will be available and that no better reservation will be available. The availability probability of a reservation can be approximated by the probability that the reservation does not violate any activity precedence constraint or capacity constraint (see section 5 for details). These probabilities can then be combined into new a priori start time and resource probability distributions, and the propagation process can be carried out all over again. The experimental results that we report in this paper have all been obtained without iterating the propagation process. We are planning to perform similar experiments with probability distributions obtained after iterating the propagation a variable number of times.

[4] In any realistic problem, Monte Carlo simulation would indeed require tremendous amounts of computations if one were to simultaneously account for all the activities and all the constraints. This is because the probability of randomly generating a schedule for all the activities that satisfy all activity precedence and resource capacity constraints is in general extremely small.

Notations

In the remainder of the paper the following notations will be used:

- $P^{PRIOR}(st_k=t)$ will denote the a priori probability that A_k will be scheduled to start at time t,
- $P^{POST}(st_k=t)$ will be the a posteriori probability that A_k starts at time t, i.e. after accounting for activity precedence constraints,
- $P_N^{POST}(st_k=t)$ represents the same probability distribution after it has been normalized to express that A_k will start exactly once,
- $D_{kij}(t)$ represents A_k's individual demand for R_{kij} at time t, and
- $D_{R_{kij}}^{aggr}(t)$ will denote the aggregate demand for R_{kij} at time t.

4. ARR: A Variable Ordering Heuristic Based on Activity Resource Reliance

ARR, the variable ordering heuristic that we study in this paper, consists in looking for the resource/time interval pair that is the most contended for and the activity that relies most on the possession of that resource over that time interval. This activity is selected as the most critical one and hence is the next one to be scheduled.

The intuition behind this heuristic is the following. If activities that critically rely on the possession of highly contended resources were not scheduled first, it is very likely that, by the time the scheduler would turn its attention to them, the reservations that are the most appropriate for these activities would no longer be available.

The aggregate demand densities introduced in subsection 3.2 are used to identify the most demanded resource/time-interval pair. The activity that contributes most to the demand for the resource over the time interval (i.e. the activity with the largest individual demand for the resource over the time interval) is interpreted as the one that relies most on the possession of that resource. Indeed the total demand of an activity A_k for one of its resource requirements R_{ki} is equal to A_k's duration and is distributed over the different alternatives, R_{kij}, for that resource, and over the different possible times when A_k can be carried out. Consequently activities with a lot of slack or several resource alternatives tend to have fairly small individual demand densities at any moment in time. They rely less on the possession of a resource at any moment in time than activities with less slack and/or fewer resource alternatives. This allows ARR to account not only for inter-order interactions but also for intra-order interactions.

The advantage of this approach lies in its relative simplicity: look for the most critical resource/time-interval pair and select the activity that relies most on it. One may however contend that this heuristic does not consider slack as an independent component to activity criticality. Instead slack is only accounted for via resource contention. Another possible problem with this heuristic is that it only considers resource reliance with respect to the most contended resource. In general an activity A_k may require several resources R_{ki}. Rigorously, A_k's criticality should therefore account for each of these resources. It should account for the contention for each of the possible alternatives R_{kij} for these resources R_{ki}, and the reliance of A_k on the possession of each of these alternatives R_{kij}. Computation of such a criticality measure is likely however to be more expensive.

5. Three Value Ordering Heuristics

In the experiments that we ran, we considered the following three value-ordering heuristics:

5.1. LCV: A Least Constraining Value Ordering Heuristic

Least constraining value ordering heuristics are known for being very good at reducing search [6, 2]. Similar heuristics have also been proposed for scheduling, even when viewed as an optimization problem. [7], for instance, suggests the use of a least constraining value ordering heuristic as a primary criterion for selecting a reservation for an

activity. The quality of the reservations is only used as a secondary criterion when there are several least constraining reservations to choose from. A similar heuristic is also outlined in [8]. The extremely small number of feasible solutions to a scheduling problem compared to the total number of schedules that one can possibly generate is what has made least constraining value ordering heuristics so attractive.

LCV is a least constraining value ordering heuristic where every reservation $\{\{st_k=t, R_{k1j_1}, R_{k2j_2}, \ldots, R_{kp_kj_{kp_k}}\}\}$, for an activity A_k, is rated according to the probability $RESERV\text{–}AVAIL(st_k=t, R_{k1j_1}, \ldots, R_{kp_kj_{p_k}})$ that it would not conflict with another activity's reservation, if one were to first schedule all the other remaining activities. The reservation with the largest such probability is interpreted as the least constraining one.

In our model, we express $RESERV\text{–}AVAIL(st_k=t, R_{k1j_1}, \ldots, R_{kp_kj_{p_k}})$ as the product of the probability that $st_k=t$ will not result in the violation of an activity precedence constraint and the conditional probability that each resource $R_{k1j_1}, R_{k2j_2}, \ldots, R_{kp_kj_{p_k}}$ will be available between t and $t+du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint :

$$RESERV\text{–}AVAIL(st_k=t, R_{k1j_1}, \ldots, R_{kp_kj_{p_k}}) = \frac{P^{POST}(st_k=t)}{P^{PRIOR}(st_k=t)} \times \prod_{R_{kij} \in \{R_{k1j_1}, \ldots, R_{kp_kj_{p_k}}\}} RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$$

where $RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$ is the conditional probability that R_{kij} will be available between t and $t+du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint.

We will approximate the (conditional) probability that R_{kij} will be available at some time τ for activity A_k with $\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)}$. When approximating $RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$, one has to be careful not to come up with too pessimistic an estimate. Indeed it is tempting to assume that the (conditional) probability that R_{kij} will be available for A_k between t and $t+du_k$ is given by the product of $\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)}$ over all possible start times τ between t and $t+du_k$. Depending on whether time is discrete or not, this product would be finite or infinite. In either case the approximation would be too pessimistic. Indeed this would be tantamount to supposing that the activities that contribute to $D^{aggr}_{R_{kij}}(\tau)$ have infinitely small durations, i.e. that these activities can possibly require R_{kij} at time τ without requiring it at $\tau-\delta\tau$ or $\tau+\delta\tau$. Instead, in order to account for the duration of these activities, we will assume that each resource R_{kij} is subdivided into a sequence of buckets of duration $AVG(du)$, where $AVG(du)$ is the average duration of the activities competing for R_{kij}. Consequently $RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$ is given by the probability that A_k can secure a number of buckets equal to its duration, i.e.:

$$RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k) = \{AVG(\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)})\}^{\frac{du_k}{AVG(du)}}$$

where $AVG(\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)})$ is simply the average of $\frac{D_{kij}(\tau)}{D^{aggr}_{R_{kij}}(\tau)}$ taken between t and $t+du_k$.

5.2. HC: A Hill-Climbing Value Ordering Heuristic

The second value ordering heuristic that we tested simply consists in ranking an activity's possible reservations according to their utilities, i.e. preferences. Reservations with the highest preferences are the first ones to be tried.

5.3. INT: An Intermediate Value Ordering Heuristic

Our third value ordering heuristic combines features from the previous two. Each reservation is rated according to the probability that it would be available and that no better reservation would be available, if one were to first schedule all the other remaining activities.

6. Preliminary Experimental Results

A testbed was implemented that allows for experimentation with a variety of variable and value ordering heuristics based on the probabilistic framework described in subsection 3.2. The system is implemented in Knowledge Craft running on top of Common Lisp, and can be run either on a MICROVAX 3200 or on a VAX 8800 under VMS.

We performed some preliminary experiments to evaluate the four heuristics presented in this paper. These experiments were run on a set of 38 scheduling problems, involving between 3 and 5 orders and a total number of activities ranging between 10 and 20. The problems involved 3 or 4 resources. They involved only activities with a unique resource requirement (R_{kl}), for which there were one or several alternatives (R_{klj}). Problems with both equally preferred and non equally preferred resource alternatives were included. The scheduling problems were built to reflect a variety of demand profiles: localized bottlenecks at the beginning, middle, and end of the problem span, global bottlenecks spanning the whole duration of the scheduling problems, and auxiliary bottlenecks were all included. Three different types of start time utility functions were allowed: all start times (between the earliest and latest start times) are equally preferred, late start times are preferred, and triangular start utility functions with a peak corresponding to the due date (minus the duration of the activity). Triangular utility functions were only assigned to the last activities of some orders. Time was discretized and a granularity equal to the third of the smallest activity duration was used. A discrete version of the formulas presented in this paper was used to compute the necessary probability distributions. The probabilities were computed using biased a priori probability distributions obtained by normalizing the utility functions over the domain of possible values of each variable. The granularity of the time intervals used for the ARR variable ordering heuristic varied from one resource to the other and was selected to be equal to the duration of the shortest activity requiring the resource.

Preliminary Experimental Results					
	RAND &HC	RAND &LCV	ARR &HC	ARR &LCV	ARR &INT
Search Efficiency	< 0.47 (> 0.27)	1.00 (0.00)	0.96 (0.05)	1.00 (0.00)	1.00 (0.00)
Schedule Value	not available	0.52 (0.08)	0.68 (0.06)	0.54 (0.06)	0.64 (0.05)

Figure 6-1: Average search efficiencies and schedule values for 5 combinations of variable and value ordering heuristics run on a preliminary set of 38 scheduling problems. The standard deviations appear between parentheses.

The experiments were measured along two dimensions: *search efficiency* (i.e. number of operations to schedule over number of search states generated) and *global utility* of the solution as defined by a normalized objective function. The normalized objective functions were built so that the best possible schedules that could be built without checking for constraint violation would have a global value of 1. There was no guarantee however that a *feasible* schedule with global value of 1 could be built. In the ideal case the search would be performed without backtracking, and the

number of search states generated would be equal to the number of activities to schedule (i.e. efficiency of 1). The quality of the schedules is more difficult to assert as the value of the objective function for the optimal schedule varied from one problem to the other and was in most cases smaller than 1. For this reason the values of the schedules should not be viewed as absolute measures. Instead they should only be used to compare the relative performances of the combinations of heuristics that we tried.

The table in Figure 6-1 reports the average search efficiencies and schedule values obtained with five combinations of variable and value ordering heuristics (for a total of 190 experiments). Standard deviations are provided between parentheses. RAND denotes a random variable ordering heuristic, where the next activity to be scheduled is selected at random from the remaining unscheduled activities. Search was stopped when it would require more than 50 search states. For RAND&HC, this cutoff rule had to be used in 12 of the 38 experiments. It did not have to be used for any of the other heuristics. The average search efficiency of RAND&HC is therefore even worse than 0.47. Because search did not terminate in almost a third of the runs with RAND&HC, no good estimate of the value of the schedules produced by this heuristic could be obtained.

7. Discussion

The results reported in Figure 6-1 clearly indicate the importance of a good variable ordering heuristic to increase search efficiency (e.g. ARR&HC vs. RAND&HC). They also indicate that a least constraining value ordering heuristic can make up for a poor variable ordering heuristic (e.g. RAND&HC vs. RAND&LCV and RAND&LCV vs. ARR&LCV). In the examples that we ran the ARR variable ordering heuristic and the LCV value ordering heuristics both contributed to limit search. The quality of the schedules produced by LCV is however very poor when compared to the other two value ordering heuristics (ARR&HC or ARR&INT vs. ARR&LCV). In particular ARR&INT performed as well as ARR&LCV as far as the efficiency of the search is concerned (neither of them had to backtrack in any of the 38 examples) but produced much better schedules. Even a simple value ordering heuristic like HC resulted in very little amount of backtracking when coupled with our variable ordering heuristic (see ARR&HC). Overall HC produced the best schedules although it required more search than INT and LCV, when coupled with ARR. There seems therefore to be a tradeoff between the amount of search performed and the quality of the resulting solution. If little time is available to come up with a solution the most promising values may be the least constraining ones as they are the least likely to result in backtracking. On the other hand, when there is more time available, one may consider looking at riskier values if they are likely to produce better solutions. A value ordering heuristic could accordingly be designed that accounts for the time available to find a schedule.

8. Concluding Remarks

In this paper, we have investigated an activity-based approach to scheduling. Because of its greater flexibility, such an approach is expected to allow for the construction of better schedules than approaches using order-based or resource-based scheduling or even combinations of the two. The price to pay for this flexibility is the potential overhead involved in the selection of the next decision point on which to focus attention. While order-based and resource-based scheduling typically require only the computation of criticality measures for each order or resource in the system, an activity-based scheduling approach may potentially require the computation of similar measures for each of the activities to be scheduled. In the simplest scenario, these measures need moreover to be recomputed every time a new activity has been scheduled. It is therefore important that the computation of these measures can be performed at a relatively cheap computational cost. One may also consider scenarios where several activities are scheduled before new criticality measures are computed.

We have presented a probabilistic framework that successively accounts for both intra-order and inter-order interactions. Although such a two-step propagation involves a slight loss of precision in the way it accounts for interactions, it has the advantage of having a relatively low computational cost [14]. More accurate probability

distributions may always be obtained by iterating the propagation process. Our probabilistic framework allows for the definition of a variety of variable and value ordering heuristics. In this paper, we have studied a simple variable-ordering heuristic, ARR, that looks for the most contended resource/time interval pair and the activity that relies the most on the possession of that resource/time interval pair. Preliminary experiments with the heuristic indicate that it greatly contributes to increasing search efficiency. Additionally our experiments with three value ordering heuristics seem to indicate that least constraining value odering heuristics such as the one advocated in [7] may not be the only viable way to maintain search at an acceptable level. Instead, in our experiments, other value ordering heuristics, when coupled with our variable ordering heuristic, produced much better schedules without significantly increasing search.

Our variable ordering heuristic is not perfect. For instance it measures activity criticality only with respect to one resource (the most contended resource/time interval pair). While the heuristic performed well in problems where each activity requires only one resource (for which there may or may not be alternatives), it may not be as effective for activities requiring several resources. We are currently looking at other variable ordering and value ordering heuristics. We are also pursuing our experiments with the heuristics presented in this paper. In particular we still have to study the behavior of these heuristics on larger scheduling problems (i.e. more than 100 activities).

Our long term interest is in the identification of a set of (texture) measures characterizing the search space, that can be used to both structure and guide search in that space. Measures of variable criticality (variable ordering heuristics) and estimates of value goodness (value ordering heuristics) are examples of such measures [4].

Acknowledgement

This research has been supported, in part, by the Defense Advance Research Projects Agency under contract #F30602-88-C-0001, and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

References

[1] J.F.Allen.
Towards a General Theory of Action and Time.
Artificial Intelligence 23(2):123-154, 1984.

[2] Rina Dechter and Judea Pearl.
Network-Based Heuristics for Constraint Satisfaction Problems.
Artificial Intelligence 34(1):1-38, 1988.

[3] M. Fox.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1983.

[4] Mark S. Fox, Norman Sadeh, and Can Baykan.
Constrained Heuristic Search.
Technical Report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1989.
Working Paper.

[5] E.C. Freuder.
A Sufficient Condition for Backtrack-free Search.
Journal of the ACM 29(1):24-32, 1982.

[6] Robert M. Haralick and Gordon L. Elliott.
Increasing Tree Search Efficiency for Constraint Satisfaction Problems.
Artificial Intelligence 14(3):263-313, 1980.

[7] N.P. Keng, D.Y.Y. Yun and M. Rossi.
Interaction-sensitive planning system for job-shop scheduling.
Expert Systems and Intelligent Manufacturing :57-69, 1988.

[8] Nicola Muscettola and Stephen Smith.
A Probabilistic Framework for Resource-Constrained Muti-Agent Planning.
In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 1063-1066. 1987.

[9] B.A. Nadel.
The General Consistent Labeling (or Constraint Satisfaction) Problem.
Technical Report DCS-TR-170, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[10] B.A. Nadel.
Three Constraint Satisfaction Algorithms and Their Complexities: Search-Order Dependent and Effectively Instance-specific Results.
Technical Report DCS-TR-171, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[11] B.A. Nadel.
Theory-based Search-order Selection for Constraint Satisfaction Problems.
Technical Report DCS-TR-183, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[12] Peng Si Ow.
Experiments in Knowledge-based Scheduling.
Technical Report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1986.

[13] Paul W. Purdom, Jr.
Search Rearrangement Backtracking and Polynomial Average Time.
Artificial Intelligence 21:117-133, 1983.

[14] N. Sadeh and M.S. Fox.
Preference Propagation in Temporal/Capacity Constraint Graphs.
Technical Report CMU-CS-88-193, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 1988.
Also appears as Robotics Institute technical report CMU-RI-TR-89-2.

[15] Stephen F. Smith and Peng Si Ow.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1013-1015. 1985.

[16] Stephen F. Smith, Peng Si Ow, Claude Lepape, Bruce McLaren, Nicola Muscettola.
Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans.
In *Proceedings 1986 SME Conference on AI in Manufacturing*, pages 123-137. 1986.

[17] Harold S. Stone and Paolo Sipala.
The average complexity of depth-first search with backtracking and cutoff.
IBM Journal of Research and Development 30(3):242-258, 1986.

REASONING UNDER UNCERTAINTY